PEMROGRAMAN JARINGAN

“Eksplorasi Mandiri 2”

DOSEN PENGAMPU:

Randi Proska Sandra, M.Sc.

OLEH:

Muhammad Alfarobi

23343011

PROGRAM STUDI INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRONIKA

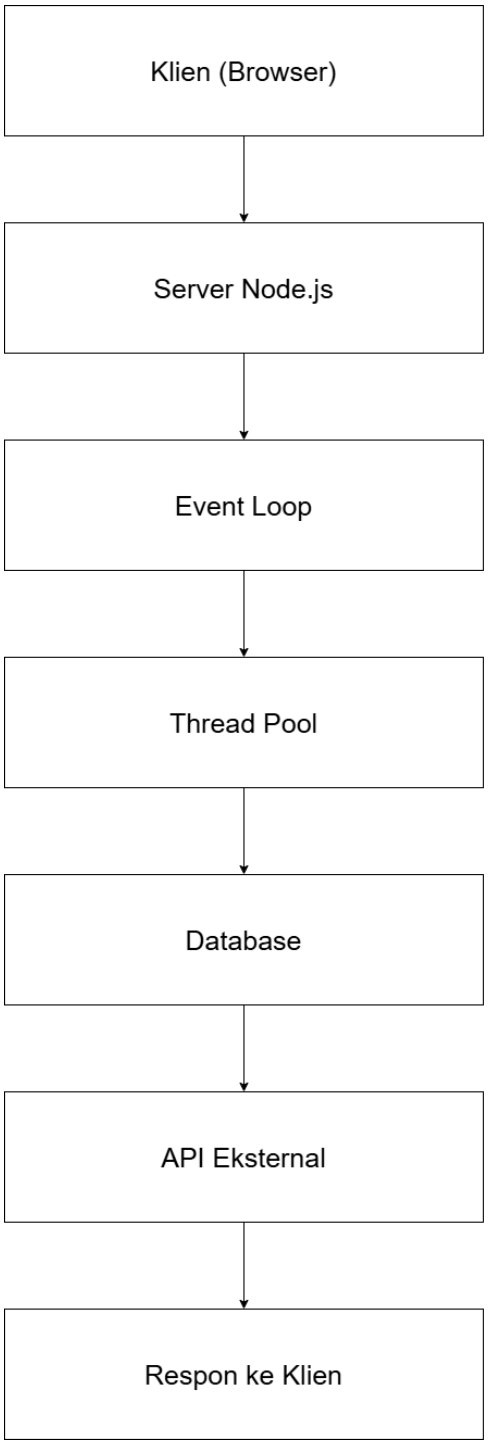
FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2025

Topik 1 – [Node.JS Application Architecture]

1. Diagram



Gambar 1. Diagram alur kerja Node.JS Application Architecture

2. Penjelasan Diagram

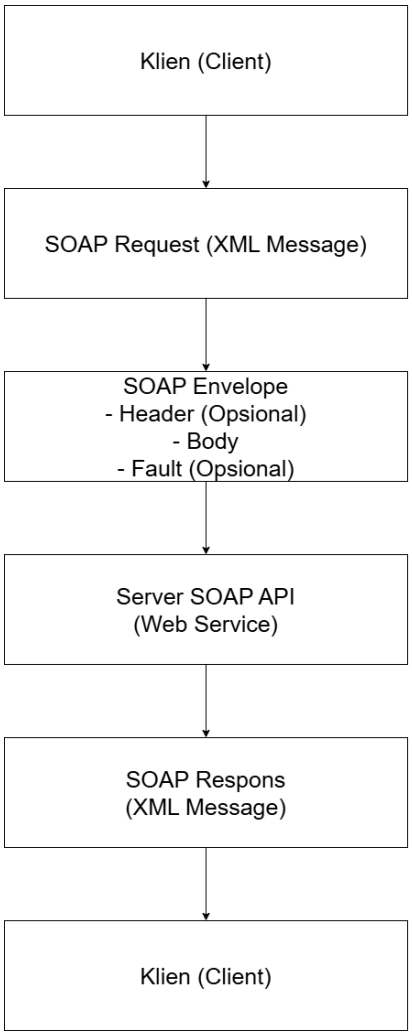
Node.js adalah lingkungan runtime JavaScript berbasis event-driven dan asynchronous yang dibangun di atas mesin V8 milik Google Chrome. Dengan arsitektur single-threaded, Node.js mampu menangani banyak koneksi secara bersamaan tanpa perlu membuat thread baru untuk setiap koneksi. Hal ini menjadikannya sangat efisien untuk aplikasi-aplikasi yang memerlukan skalabilitas tinggi dan latensi rendah, seperti aplikasi real-time dan I/O-intensive.

Alur Kerja:

1. Klien mengirimkan permintaan: Pengguna mengakses aplikasi melalui browser atau aplikasi mobile, mengirimkan permintaan HTTP ke server Node.js.
2. Server Node.js menerima permintaan: Server menerima permintaan dan menambahkannya ke dalam event queue.
3. Event Loop memproses permintaan: Event loop memeriksa event queue dan mengeksekusi callback untuk permintaan yang telah siap diproses.
4. Thread Pool menangani operasi blocking: Untuk operasi yang bersifat blocking, seperti akses ke sistem file atau database, thread pool digunakan untuk menanganinya secara paralel.
5. Interaksi dengan Database: Server berinteraksi dengan database untuk mengambil atau menyimpan data sesuai dengan permintaan klien.
6. Integrasi dengan API Eksternal: Jika diperlukan, server dapat berkomunikasi dengan layanan pihak ketiga untuk mendapatkan data atau fungsionalitas tambahan.
7. Respons dikirim ke Klien: Setelah pemrosesan selesai, server mengirimkan respons kembali ke klien, biasanya dalam format JSON atau HTML.

Topik 2 – [API Architecture Patterns (SOAP)]

1. Diagram



Gambar 2. Diagram alur kerja API Architecture Patterns (SOAP)

2. Penjelasan Diagram

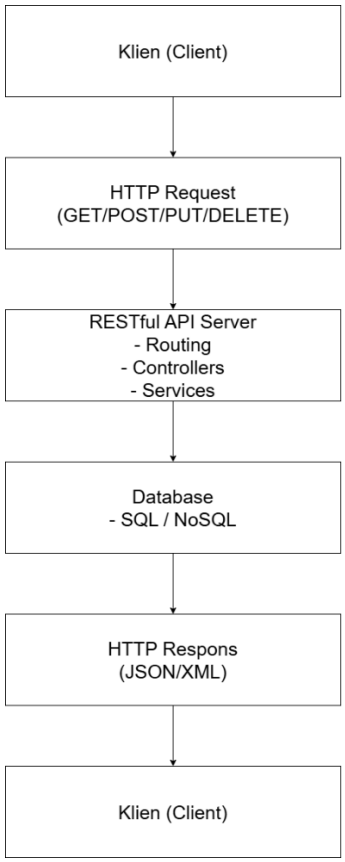
SOAP (Simple Object Access Protocol) adalah protokol komunikasi berbasis XML yang digunakan untuk pertukaran data antar aplikasi melalui jaringan, seperti HTTP atau SMTP. SOAP dirancang untuk mendukung komunikasi yang aman, andal, dan terstruktur, menjadikannya pilihan utama dalam sistem enterprise yang memerlukan transaksi kompleks dan tingkat keamanan tinggi.

Alur Kerja

1. Klien Mengirim Permintaan: Klien membuat pesan SOAP yang berisi permintaan, dikemas dalam format XML, dan mengirimkannya ke server melalui protokol transportasi yang sesuai (misalnya HTTP).
2. SOAP Envelope: Pesan SOAP dibungkus dalam elemen Envelope, yang terdiri dari:
   - Header: Opsional, berisi informasi tambahan seperti otentikasi atau routing.
   - Body: Berisi payload utama, yaitu permintaan atau respons.
   - Fault: Opsional, digunakan untuk menyampaikan informasi kesalahan jika terjadi.
3. Server Memproses Permintaan: Server menerima pesan SOAP, memprosesnya sesuai dengan logika bisnis yang ditentukan, dan menghasilkan respons.
4. SOAP Response: Server mengirimkan respons kembali ke klien dalam format SOAP yang serupa, berisi hasil dari permintaan yang diajukan.
5. Klien Menerima Respons: Klien menerima dan memproses respons SOAP, kemudian menampilkan hasilnya kepada pengguna atau melanjutkan proses lainnya.

Topik 3 – [API Architecture Patterns (RESTful)]

1. Diagram



Gambar 3. Diagram alur kerja API Architecture Patterns (RESTful)

2. Penjelasan Diagram

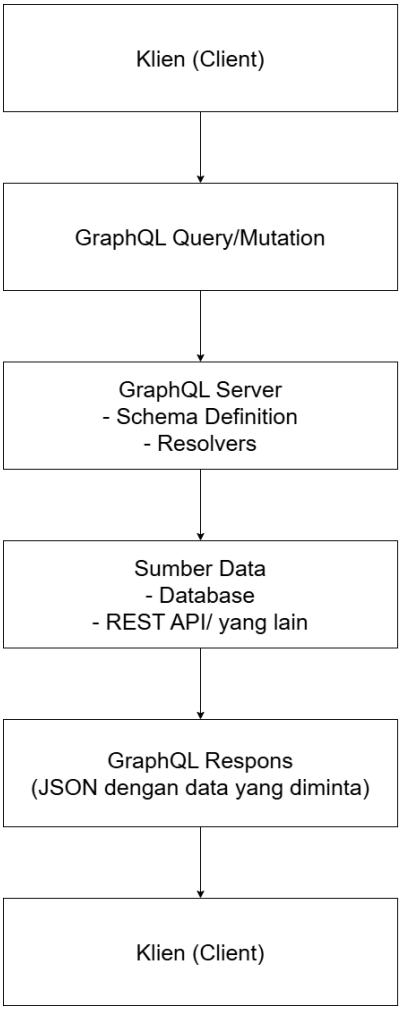
RESTful API (Representational State Transfer) adalah gaya arsitektur untuk membangun layanan web yang memanfaatkan protokol HTTP untuk pertukaran data antara klien dan server. REST menekankan pada stateless communication, artinya setiap permintaan dari klien ke server harus mengandung semua informasi yang diperlukan untuk memproses permintaan tersebut tanpa menyimpan state di server.

Alur Kerja

1. Klien Mengirim Permintaan: Klien mengakses RESTful API dengan HTTP request ke endpoint tertentu, menggunakan metode HTTP sesuai kebutuhan.
2. Server Menerima dan Memproses Permintaan: RESTful API server menggunakan routing untuk menyalurkan permintaan ke controller dan service yang sesuai.
3. Interaksi dengan Database: Server mengambil, menyimpan, memperbarui, atau menghapus data di database sesuai permintaan.
4. Respons ke Klien: Server mengirimkan HTTP response, biasanya dalam format JSON atau XML, yang berisi data atau status dari permintaan.
5. Klien Menerima Respons: Klien memproses data yang diterima dan menampilkannya kepada pengguna atau melanjutkan alur bisnis.

Topik 4 – [API Architecture Patterns (GraphQL)]

1. Diagram



Gambar 4. Diagram alur kerja API Architecture Patterns (GraphQL)

2. Penjelasan Diagram

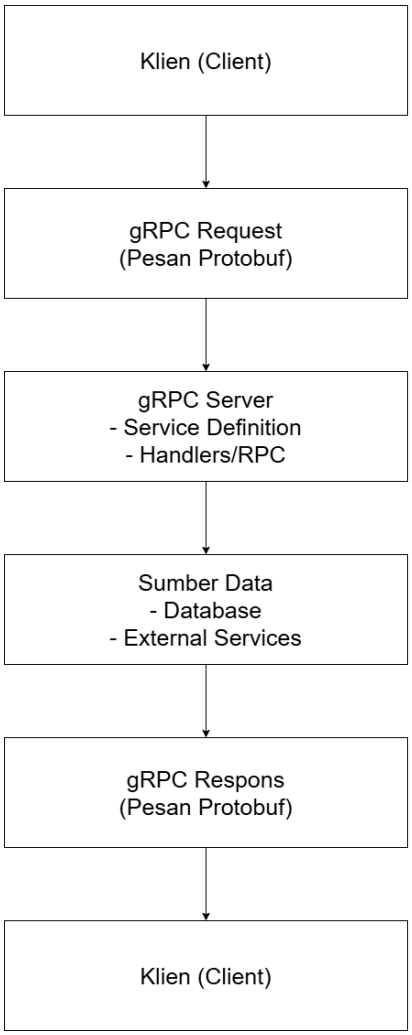
GraphQL adalah bahasa query untuk API dan runtime untuk mengeksekusi query tersebut terhadap data yang tersedia. Dikembangkan oleh Facebook, GraphQL memungkinkan klien untuk meminta data secara spesifik sesuai kebutuhan, berbeda dengan REST yang biasanya mengembalikan seluruh resource.

Alur Kerja

1. Klien Mengirim Query atau Mutation: Klien membuat request GraphQL yang berisi query (mengambil data) atau mutation (mengubah data) ke server.
2. Server Memproses Query: GraphQL server memeriksa query terhadap skema yang telah didefinisikan dan mengeksekusi resolver untuk tiap field yang diminta.
3. Mengambil Data dari Sumber Data: Resolver mengakses berbagai sumber data, seperti database, REST API lain, atau layanan eksternal.
4. Menyusun Respons: Server membentuk respons dalam format JSON, hanya berisi data yang diminta oleh klien.
5. Klien Menerima Data: Klien memproses data sesuai kebutuhan, mengurangi over-fetching dan under-fetching dibanding REST.

Topik 5 – [API Architecture Patterns (gRPC)]

1. Diagram



Gambar 5. Diagram alur kerja API Architecture Patterns (gRPC)

2. Penjelasan Diagram

gRPC (gRPC Remote Procedure Call) adalah framework open-source yang dikembangkan oleh Google untuk melakukan komunikasi antar layanan (service-to-service) dengan performa tinggi. gRPC menggunakan Protocol Buffers (Protobuf) sebagai format serialisasi data, yang lebih cepat dan efisien dibanding JSON atau XML.

Alur Kerja

1. Klien Mengirim Request: Klien membuat panggilan RPC ke server gRPC, mengirimkan request yang telah dikemas dalam format Protobuf.
2. Server Memproses Request: gRPC server memeriksa request terhadap definisi service yang ada, lalu mengeksekusi handler atau metode RPC terkait.
3. Mengakses Sumber Data: Server mengambil data dari database atau layanan eksternal sesuai kebutuhan.
4. Menyusun Respons: Server membalas request dengan Protobuf message berisi data atau hasil eksekusi RPC.
5. Klien Menerima Response: Klien menerima response dan memprosesnya sesuai kebutuhan aplikasi, dengan latency rendah dan efisiensi tinggi.

REFERENSI

- https://nodejs.org/en/about
- https://www.w3schools.com/nodejs/nodejs_architecture.asp
- https://www.geeksforgeeks.org/node-js/node-js-web-application-architecture/
- https://medium.com/@komalpal/soap-api-architectural-style-9309873df6
- https://www.ibm.com/docs/en/cics-ts/6.x?topic=format-soap-web-services-architecture
- https://blog.postman.com/soap-api-definition/
- https://restfulapi.net/
- https://www.ibm.com/docs/en/cloud-paks/cp4i/2022.2?topic=services-restful-api-architecture
- https://www.geeksforgeeks.org/restful-web-services/
- https://graphql.org/learn/
- https://www.apollographql.com/docs/graphql/
- https://www.geeksforgeeks.org/graphql/
- https://grpc.io/docs/what-is-grpc/introduction/
- https://www.baeldung.com/gRPC
- https://medium.com/geekculture/what-is-grpc-architecture-and-its-features-3d1cb21a4c06